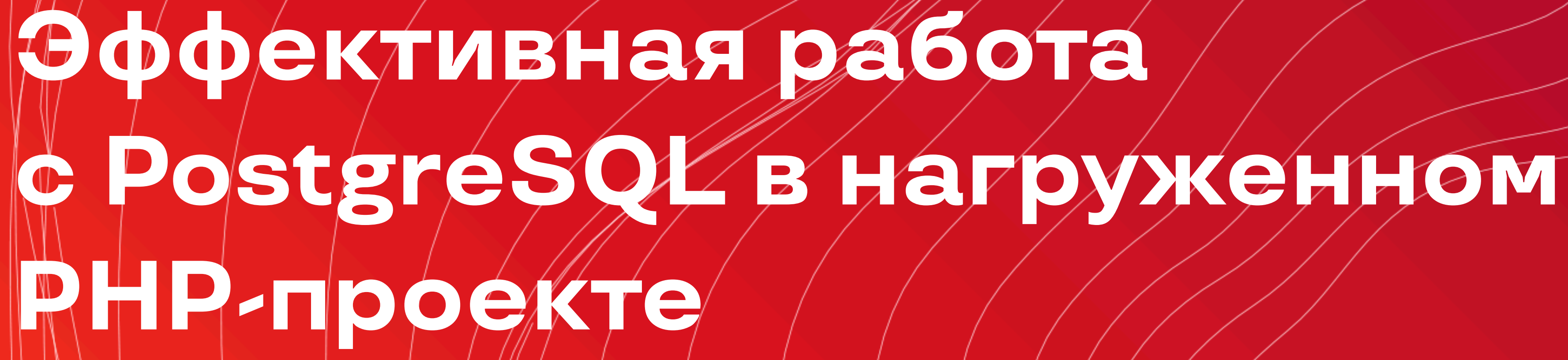


Эффективная работа с PostgreSQL в нагруженном PHP-проекте

Салихов Ильяс

RetailCRM





- B2B SaaS
- Автоматизация eCommerce и ритейла
- >3000 клиентов в России и за рубежом
- 30% интернет-магазинов в России
- Команда разработки >50 человек





- ~250 bare metal
- >50 серверов БД
- ~100 000 tps



О чем будем говорить

- Подключение к PostgreSQL
- Управление схемой данных
- Миграции
- Индексы



Пара терминов в докладе

- **Модель данных** — структура БД, описанная в приложении (на уровне моделей Doctrine)
- **Схема данных** — структура в PostgreSQL (таблицы, поля, констрейнты и т.п.)



Подключение к PostgreSQL



Connection pooler



PHP → PostgreSQL

HTTP

Requests

Database

Connections

PHP-FPM

PostgreSQL



PHP → PostgreSQL

```
<?php

$pgConnection = Doctrine\DBAL\DriverManager::getConnection([
    'dbname' => 'demo',
    'driver' => 'pdo_pgsql',
    'host'   => 'postgres_host',
    'port'   => 5432,
]);

$statement = $pgConnection->executeQuery('SELECT first_name FROM user LIMIT 100');
```




PHP → PostgreSQL

HTTP

Requests

Database

Connections

PHP-FPM

PostgreSQL



PHP → Pgbouncer → PostgreSQL

HTTP

Requests

Pgbouncer

Connections

Database

Connections

PHP-FPM

Pgbouncer

PostgreSQL



PHP → Pgbouncer → PostgreSQL

```php
<?php

$pgConnection = Doctrine\DBAL\DriverManager::getConnection([
    'dbname' => 'demo',
    'driver' => 'pdo_pgsql',
    'host'   => 'pgbouncer_host',
    'port'   => 6432,
]);

$statement = $pgConnection->executeQuery('SELECT first_name FROM user LIMIT 100');
```




Эмуляция регулярных коннектов к БД

```php
<?php

$stopwatch = new Symfony\Component\Stopwatch\Stopwatch(true);
$connectionParams = [ /* ... */ ];

for ($i = 0; $i < 500; $i++) {
    $pgConnection = Doctrine\DBAL\DriverManager::getConnection($connectionParams);

    $stopwatch->start('dbal', 'dbal');
    $pgConnection->executeQuery('SELECT first_name FROM user LIMIT 100');
    $stopwatch->stop('dbal');

    $stopwatch->reset();
    $pgConnection->close();
}
```




PHP+Pg vs PHP+Pgbouncer+Pg

DBAL PostgreSQL	2,58 ms	
DBAL Pgbouncer	**0,50 ms**	**x5**
CLI PostgreSQL	0,41 ms	

```
demo=> SELECT first_name FROM user LIMIT 100;
...
(100 строк)

Время: 0,412 мс
```




Architecture: PHP + Pgbouncer + Pg

Вариант 1



Architecture: PHP + Pgbouncer + Pg

Вариант 2

HTTP

PHP-FPM

Pgbouncer

PostgreSQL

App server

DB server



Architecture: PHP + Pgbouncer + Pg

Вариант 1

\+ Точка коннекта ближе к приложению

\+ Индивидуальные настройки pgbouncer

– Сложность конфигурирования

– Контроль за общим числом коннектов к СУБД



Аналоги pgbouncer

Odyssey от Яндекс https://github.com/yandex/odyssey

pgagroal https://github.com/agroal/pgagroal



PHP → Pgbouncer → PostgreSQL

+ Переиспользование коннектов к PostgreSQL

+ Ускорение SQL-запросов (особенно легких)

+ Экономия ресурсов СУБД на создание connections

+ Контролируемый «барьер» при резком увеличении нагрузки



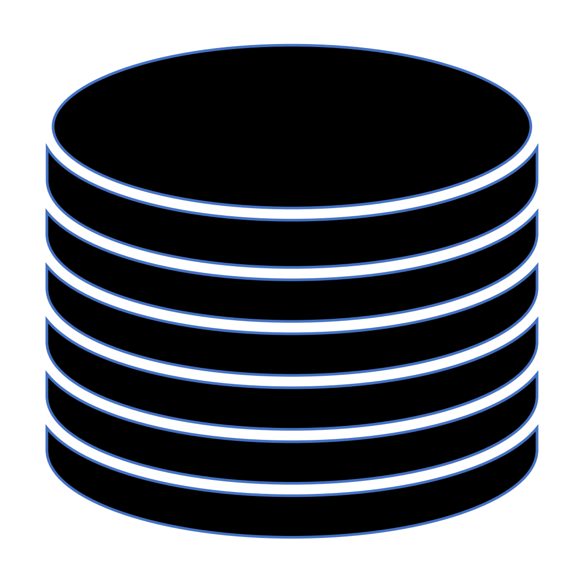
Timeouts



Медленный SQL-запрос

```
max_execution_time =
60 sec
```

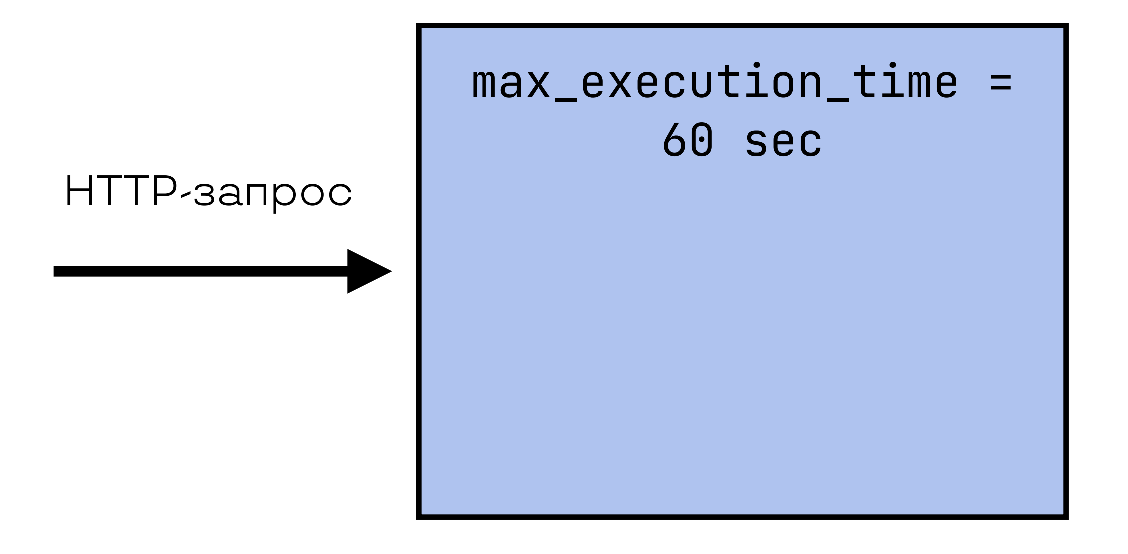
PHP-FPM

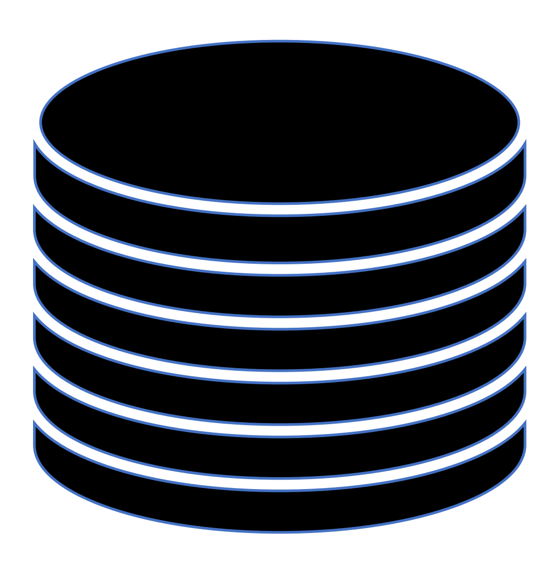


PostgreSQL



Медленный SQL-запрос







Медленный SQL-запрос

HTTP-запрос

```
max_execution_time =
60 sec
```

SQL-запрос

PHP-FPM

PostgreSQL



Медленный SQL-запрос

max_execution_time =
60 sec

HTTP 504

Долгий SQL-запрос

PHP-FPM

PostgreSQL



Медленный SQL-запрос

HTTP 504

HTTP 504

HTTP 504

max_execution_time = 60 sec

PHP-FPM

Долгий SQL-запрос

Долгий SQL-запрос

Долгий SQL-запрос

PostgreSQL



Тайм-аут выполнения запросов

```
SET statement_timeout = 70000
```




Тайм-аут выполнения запросов

```yaml
# config/packages/doctrine.yaml

doctrine:
    dbal:
        url: '%env(resolve:DATABASE_URL)%'
        options:
            statement_timeout: '%env(resolve:PSQL_STATEMENT_TIMEOUT)%'
```




Тайм-аут выполнения запросов

```php
final class AppDriver implements Driver
{
    public function connect(array $params)
    {
        $driverOptions = $params['driverOptions'] ?? [];

        // ...

        $connection = new Connection($pdo);

        $connection->exec(sprintf(
            'SET statement_timeout = %d',
            $driverOptions['statement_timeout'] ?? 0,
        ));

        return $connection;
    }
```




Тайм-аут выполнения запросов

Для CLI в `.env` ставим более высокие таймауты



Нельзя выставлять в PostgreSQL

Выставляем **только на уровне** приложения

Из документации PostgreSQL

> Setting statement_timeout in postgresql.conf **is not recommended** because it would affect all sessions.



Тайм-аут локов запросов

```
SET LOCAL lock_timeout = 10000
```

Выставляем:

1. или в рамках connection
2. или на уровне транзакций



Управление схемой данных и миграции



Управление схемой в Doctrine

- Создали/изменили модель

```
php bin/console make:entity Product
```




Управление схемой в Doctrine

- Создали/изменили модель

```
php bin/console make:entity Product
```

- Сгенерировали миграцию

```
php bin/console make:migration
```




Управление схемой в Doctrine

- Создали/изменили модель

```
php bin/console make:entity Product
```

- Сгенерировали миграцию

```
php bin/console make:migration
```

- Применили миграцию

```
php bin/console doctrine:migration:migrate
```




Управление схемой в Doctrine

- Создали/изменили модель

```
php bin/console make:entity
```

- Сгенерировали миграцию

```
php bin/console make:migration
```

- Применили миграцию

```
php bin/console doctrine:migration:migrate
```





...но есть нюансы



Управление схемой в Doctrine

1. Миграции требуется дорабатывать



Управление схемой в Doctrine

1. Миграции требуется дорабатывать
2. Где доработки, там и ошибки



Управление схемой в Doctrine

1. Миграции требуется дорабатывать
2. Где доработки, там и ошибки
3. В итоге модель данных расходится со схемой данных



Контроль соответствия модели и схемы данных



doctrine:schema:update

```
$ php bin/console doctrine:schema:update --dump-sql

CREATE TABLE product ...

$ php bin/console doctrine:schema:update --dump-sql

 [OK] Nothing to update - your database is already in sync with the
current entity metadata.
```




Хелпер проверки схемы в Makefile

```
db-check-schema-diff:
	@DB_DIFF=`php bin/console doctrine:schema:update --dump-sql`; \
		case "$$DB_DIFF" in \
			*'your database is already in sync'*) \
			echo 'Your database is already in sync with the metadata'; \
			exit 0; \
			;; \
		esac; echo "$$DB_DIFF"; exit 1;
```




Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
```




Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
# проверяем, что схема соответствует модели
$ make db-check-schema-diff
```




Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
# проверяем, что схема соответствует модели
$ make db-check-schema-diff

# проигрываем до первой (down) и снова до последней (up)
$ php bin/console doctrine:migrations:migrate -q 00000000000000
$ php bin/console doctrine:migrations:migrate -q
$ make db-check-schema-diff
```




Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
# проверяем, что схема соответствует модели
$ make db-check-schema-diff

# проигрываем до первой (down) и снова до последней (up)
$ php bin/console doctrine:migrations:migrate -q 00000000000000
$ php bin/console doctrine:migrations:migrate -q
$ make db-check-schema-diff

# валидируем средствами доктрины
$ php app/console doctrine:schema:validate
```




Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
# проверяем, что схема соответствует модели
$ make db-check-schema-diff

# проигрываем до первой (down) и снова до последней (up)
$ php bin/console doctrine:migrations:migrate -q 00000000000000
$ php bin/console doctrine:migrations:migrate -q
$ make db-check-schema-diff

# валидируем средствами доктрины
$ php app/console doctrine:schema:validate
```

Add to CI



Контроль схемы данных

```
# создаем базу и схему данных через миграции
$ php bin/console doctrine:database:create -n
$ php bin/console doctrine:migration:migrate -q
# проверяем, что схема соответствует модели
$ make db-check-schema-diff

# проигрываем до первой (down) и снова до последней (up)
$ php bin/console doctrine:migrations:migrate -q 00000000000000
$ php bin/console doctrine:migrations:migrate -q
$ make db-check-schema-diff

# валидируем средствами доктрины
$ php app/console doctrine:schema:validate
```

Add to CI

Profit!



Миграции



Миграции
Кейс 1. Заполнение колонки



Модель Product

```
phprussia2022=# \d product
                              Table "public.product"
  Column  |          Type          | Collation | Nullable |          Default
----------+------------------------+-----------+----------+-------------------------
 id       | integer                |           | not null |
 name     | character varying(255) |           | not null |
 price    | numeric(10,2)          |           | not null |
 currency | character varying(10)  |           |          | NULL::character varying
Indexes:
    "product_pkey" PRIMARY KEY, btree (id)
```




Модель Product. 100 mln rows

```
phprussia2022=# select count(*) from product;
   count
------------
100 000 000
(1 row)
```




Делаем currency not null

```
--- a/src/Entity/Product.php
+++ b/src/Entity/Product.php
@@ -28,7 +28,7 @@ class Product
     private $price;

     /**
-     * @ORM\Column(type="string", length=10, nullable=true)
+     * @ORM\Column(type="string", length=10)
      */
     private $currency;
```

Делаем currency not null

```
final class Version20221025000000 extends AbstractMigration
{
    public function up(Schema $schema): void
    {
        $this->addSql('ALTER TABLE product ALTER currency SET NOT NULL');
    }

    public function down(Schema $schema): void
    {
        $this->addSql('ALTER TABLE product ALTER currency DROP NOT NULL');
    }
}
```




Делаем currency not null

```
$ php bin/console doctrine:migrations:migrate -n
[notice] Migrating up to DoctrineMigrations\Version20221025145525

In Connection.php line 69:

  SQLSTATE[23502]: Not null violation: 7 ERROR:  column "currency"
contains null values
```

Проблема 🤔



Самый очевидный фикс

```php
final class Version20221025000000 extends AbstractMigration
{
    public function up(Schema $schema): void
    {
        $this→addSql('UPDATE product SET currency = ? WHERE currency IS NULL', ['USD']);
        $this→addSql('ALTER TABLE product ALTER currency SET NOT NULL');
    }

    public function down(Schema $schema): void
    {
        $this→addSql('ALTER TABLE product ALTER currency DROP NOT NULL');
    }
}
```




Проблемы реализации

1. В таблице 100 млн записей



Проблемы реализации

1. В таблице 100 млн записей
2. Миграция выполняется в транзакции



Проблемы реализации

1. В таблице 100 млн записей
2. Миграция выполняется в транзакции
3. Лок миграцией рантайм-запросов к БД



1. Делим миграцию на две

```
final class Version20221025000001 extends AbstractMigration
{
    public function up(Schema $schema): void
    {
        $this->addSql('UPDATE product SET currency = ? WHERE currency IS NULL', ['USD']);
    }
}
```




1. Делим миграцию на две

```
final class Version20221025000002 extends AbstractMigration
{
    public function up(Schema $schema): void
    {
        $this->addSql('ALTER TABLE product ALTER currency SET NOT NULL');
    }

    public function down(Schema $schema): void
    {
        $this->addSql('ALTER TABLE product ALTER currency DROP NOT NULL');
    }
}
```




2. Выключаем транзакционность

```
final class Version20221025000001 extends AbstractMigration
{
    public function up(Schema $schema): void
    {
        $this->addSql(
            'UPDATE product SET currency = ? WHERE currency IS NULL', ['USD']
        );
    }

    public function isTransactional(): bool
    {
        return false;
    }
}
```




3. Обновляем порциями

```
public function up(Schema $schema): void
{
    $maxId = $this->connection->fetchOne('SELECT max(id) FROM product WHERE currency IS NULL');
    if (null === $maxId) {
        return;
    }

    $cursor = 0;
    while ($cursor <= $maxId) {
        $this->addSql(
            'UPDATE product SET currency = ? WHERE id > ? AND id <= ? AND currency IS NULL',
            ['USD', $cursor, $cursor + self::LIMIT < $maxId ? $cursor + self::LIMIT : $maxId]
        );

        $cursor += self::LIMIT;
    }

    $this->addSql(
        'UPDATE product SET currency = ? WHERE id > ? AND currency IS NULL',
        ['USD', $maxId]
    );
}
```




3. Обновляем порциями

```php
public function up(Schema $schema): void
{
    $maxId = $this->connection->fetchOne('SELECT max(id) FROM product WHERE currency IS NULL');
    if (null === $maxId) {
        return;
    }

    $cursor = 0;
    while ($cursor <= $maxId) {
        $this->addSql(
            'UPDATE product SET currency = ? WHERE id > ? AND id <= ? AND currency IS NULL',
            ['USD', $cursor, $cursor + self::LIMIT < $maxId ? $cursor + self::LIMIT : $maxId]
        );

        $cursor += self::LIMIT;
    }

    $this->addSql(
        'UPDATE product SET currency = ? WHERE id > ? AND currency IS NULL',
        ['USD', $maxId]
    );
}
```




3. Обновляем порциями

```php
public function up(Schema $schema): void
{
    $maxId = $this->connection->fetchOne('SELECT max(id) FROM product WHERE currency IS NULL');
    if (null === $maxId) {
        return;
    }

    $cursor = 0;
    while ($cursor <= $maxId) {
        $this->addSql(
            'UPDATE product SET currency = ? WHERE id > ? AND id <= ? AND currency IS NULL',
            ['USD', $cursor, $cursor + self::LIMIT < $maxId ? $cursor + self::LIMIT : $maxId]
        );

        $cursor += self::LIMIT;
    }

    $this->addSql(
        'UPDATE product SET currency = ? WHERE id > ? AND currency IS NULL',
        ['USD', $maxId]
    );
}
```

3. Обновляем порциями

```php
public function up(Schema $schema): void
{
    $maxId = $this->connection->fetchOne('SELECT max(id) FROM product WHERE currency IS NULL');
    if (null === $maxId) {
        return;
    }

    $cursor = 0;
    while ($cursor <= $maxId) {
        $this->addSql(
            'UPDATE product SET currency = ? WHERE id > ? AND id <= ? AND currency IS NULL',
            ['USD', $cursor, $cursor + self::LIMIT < $maxId ? $cursor + self::LIMIT : $maxId]
        );

        $cursor += self::LIMIT;
    }

    $this->addSql(
        'UPDATE product SET currency = ? WHERE id > ? AND currency IS NULL',
        ['USD', $maxId]
    );
}
```




3. Обновляем порциями по id

```
phprussia2022=# explain UPDATE product SET currency = 'USD' WHERE
phprussia2022-#     id > 50000 AND id ⩽ 60000 AND currency IS NULL;
                                   QUERY PLAN
-------------------------------------------------------------------------------------
 Update on product  (cost=429.33..3100.73 rows=7943 width=87)
   →  Bitmap Heap Scan on product  (cost=429.33..3100.73 rows=7943 width=87)
        Recheck Cond: ((id > 50000) AND (id ⩽ 60000))
        Filter: (currency IS NULL)
        →  Bitmap Index Scan on product_pkey  (cost=0.00..427.35 rows=9893 width=0)
              Index Cond: ((id > 50000) AND (id ⩽ 60000))
 Planning Time: 0.071 ms
 Execution Time: 1.298 ms
```




3. Обновляем порциями через LIMIT/OFFSET

```
phprussia2022=# EXPLAIN UPDATE product SET currency = 'USD' WHERE id IN (
phprussia2022-#   SELECT id FROM product WHERE currency IS NULL ORDER BY id ASC LIMIT 10000 OFFSET 50000
phprussia2022-# );
                                                      QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
 Update on product  (cost=4643.84..8541.87 rows=10000 width=114)
   ->  Hash Semi Join  (cost=4643.84..8541.87 rows=10000 width=114)
         Hash Cond: (product.id = "ANY_subquery".id)
         ->  Seq Scan on product  (cost=0.00..3524.01 rows=100101 width=48)
         ->  Hash  (cost=4518.84..4518.84 rows=10000 width=32)
               ->  Subquery Scan on "ANY_subquery"  (cost=3682.44..4518.84 rows=10000 width=32)
                     ->  Limit  (cost=3682.44..4418.84 rows=10000 width=4)
                           ->  Index Scan using product_pkey on product product_1  (cost=0.42..7371.89 rows=100101 width=4)
                                 Filter: (currency IS NULL)
 Planning Time: 0.502 ms
 Execution Time: 83.349 ms
```




Как результат

1. Не мешаем runtime приложения lock'ами при деплое



Как результат

1. Не мешаем runtime приложения lock'ами при деплое
2. Можно поэтапно деплоить каждую миграцию



Как вести себя приложению во время наката миграции?



Миграции
Кейс 2. Миграция 1-1 в 1-N



Делаем множество цен у товаров

product	
id	serial
name	string
price	decimal
currency	string



Делаем множество цен у товаров

product	
id	serial
name	string

product_price	
id	serial
product_id	int
price	decimal
currency	string



Этапы миграции. 1 этап

product	
id	serial
name	string
price	decimal
currency	string

product_price	
id	serial
product_id	int
price	decimal
currency	string



Этапы миграции. 2 этап

product	
id	serial
name	string
price	decimal
currency	string

product_price	
id	serial
product_id	int
price	decimal
currency	string



Этапы миграции. 3 этап

product	
id	serial
name	string
~~price~~	~~decimal~~
~~currency~~	~~string~~

product_price	
id	serial
product_id	int
price	decimal
currency	string



В виде одной миграции

```
public function up(Schema $schema): void
{
    $this->addSql(<<<SQL
        CREATE TABLE product_price (
            id SERIAL NOT NULL,
            product_id INT NOT NULL,
            price NUMERIC(10, 2) NOT NULL,
            currency VARCHAR(10) NOT NULL,
            PRIMARY KEY(id)
        )
    SQL);
    $this->addSql('CREATE INDEX IDX_6B94598545846650A ON product_price (product_id)');
    $this->addSql(<<<SQL
        ALTER TABLE product_price
            ADD CONSTRAINT FK_6B94598545846650A
            FOREIGN KEY (product_id)
            REFERENCES product (id)
            ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE
    SQL);
    $this->addSql('INSERT INTO product_price (product_id, price, currency) SELECT id, price, currency FROM product');
    $this->addSql('ALTER TABLE product DROP price');
    $this->addSql('ALTER TABLE product DROP currency');
}
```




В виде одной миграции

```
public function up(Schema $schema): void
{
    $this->addSql(<<<SQL
        CREATE TABLE product_price (
            id SERIAL NOT NULL,
            product_id INT NOT NULL,
            price NUMERIC(10, 2) NOT NULL,
            currency VARCHAR(10) NOT NULL,
            PRIMARY KEY(id)
        )
    SQL);
    $this->addSql('CREATE INDEX IDX_6B9459854584665A ON product_price (product_id)');
    $this->addSql(<<<SQL
        ALTER TABLE product_price
            ADD CONSTRAINT FK_6B9459854584665A
            FOREIGN KEY (product_id)
            REFERENCES product (id)
            ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE
    SQL);
    $this->addSql('INSERT INTO product_price (product_id, price, currency) SELECT id, price, currency FROM product');
    $this->addSql('ALTER TABLE product DROP price');
    $this->addSql('ALTER TABLE product DROP currency');
}
```




В виде одной миграции

```
public function up(Schema $schema): void
{
    $this->addSql(<<<SQL
        CREATE TABLE product_price (
            id SERIAL NOT NULL,
            product_id INT NOT NULL,
            price NUMERIC(10, 2) NOT NULL,
            currency VARCHAR(10) NOT NULL,
            PRIMARY KEY(id)
        )
    SQL);
    $this->addSql('CREATE INDEX IDX_6B9459854584665A ON product_price (product_id)');
    $this->addSql(<<<SQL
        ALTER TABLE product_price
            ADD CONSTRAINT FK_6B9459854584665A
            FOREIGN KEY (product_id)
            REFERENCES product (id)
            ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE
    SQL);
    $this->addSql('INSERT INTO product_price (product_id, price, currency) SELECT id, price, currency FROM product');
    $this->addSql('ALTER TABLE product DROP price');
    $this->addSql('ALTER TABLE product DROP currency');
}
```




В виде одной миграции

```
public function up(Schema $schema): void
{
    $this→addSql(<<<SQL
        CREATE TABLE product_price (
            id SERIAL NOT NULL,
            product_id INT NOT NULL,
            price NUMERIC(10, 2) NOT NULL,
            currency VARCHAR(10) NOT NULL,
            PRIMARY KEY(id)
        )
    SQL);
    $this→addSql('CREATE INDEX IDX_6B9459854584665A ON product_price (product_id)');
    $this→addSql(<<<SQL
        ALTER TABLE product_price
            ADD CONSTRAINT FK_6B9459854584665A
            FOREIGN KEY (product_id)
            REFERENCES product (id)
            ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE
    SQL);
    $this→addSql('INSERT INTO product_price (product_id, price, currency) SELECT id, price, currency FROM product');
    $this→addSql('ALTER TABLE product DROP price');
    $this→addSql('ALTER TABLE product DROP currency');
}
```




Вводные

1. Большая таблица товаров ~100 млн записей



Вводные

1. Большая таблица товаров ~100 млн записей
2. Постоянная нагрузка ~100 000 tps к базе



Вводные

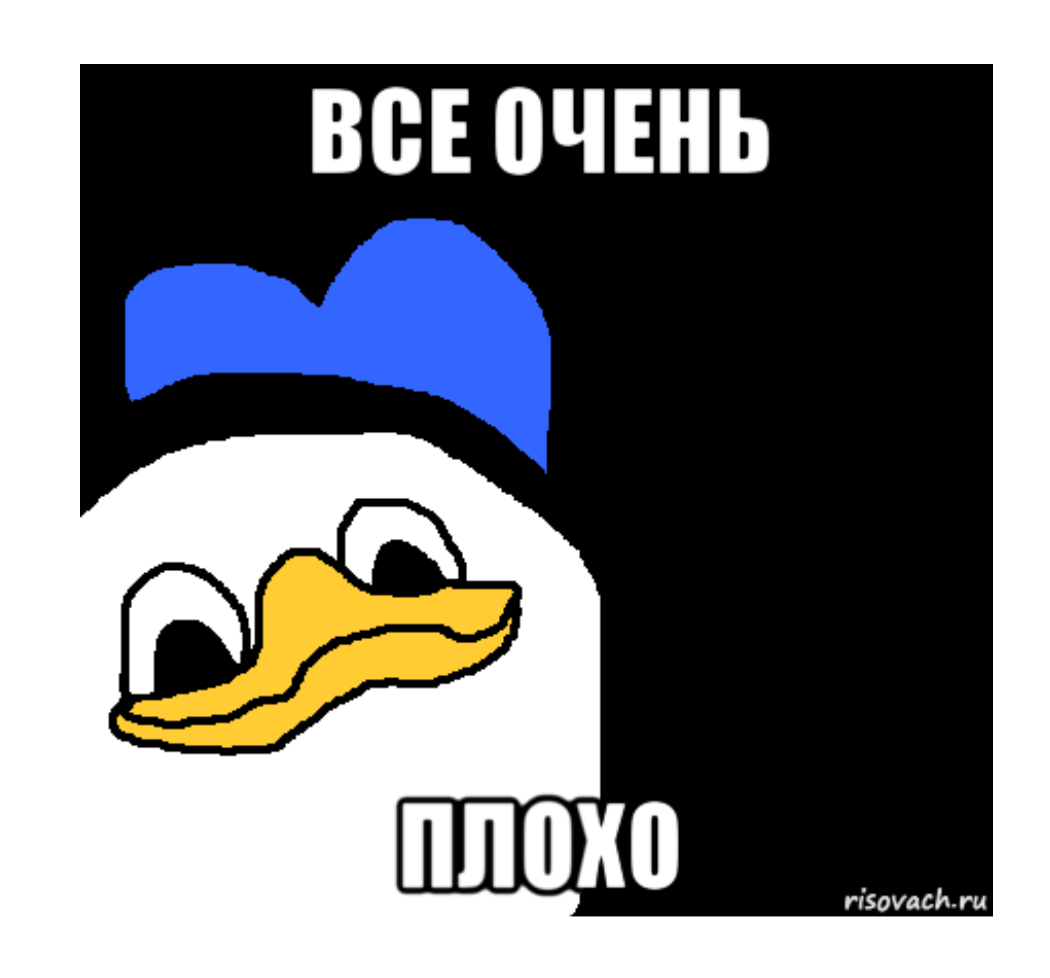
1. Большая таблица товаров ~100 млн записей
2. Постоянная нагрузка ~100 000 tps к базе
3. Минимизировать локи при внедрении



Вводные

1. Большая таблица товаров ~100 млн записей
2. Постоянная нагрузка ~100 000 tps к базе
3. Минимизировать локи при внедрении
4. Обеспечить работу во время внедрения





Локи, отказы приложения и 500-ые ;%!!?((



Разделим на несколько миграций и перенесем данные порциями! 🤓



Есть проблемки

1. Да, снижается вероятность локов



Есть проблемки

1. Да, снижается вероятность локов
2. Но длительность внедрения увеличивается



Есть проблемки

1. Да, снижается вероятность локов
2. Но длительность внедрения увеличивается
3. А если внедрение длится час?



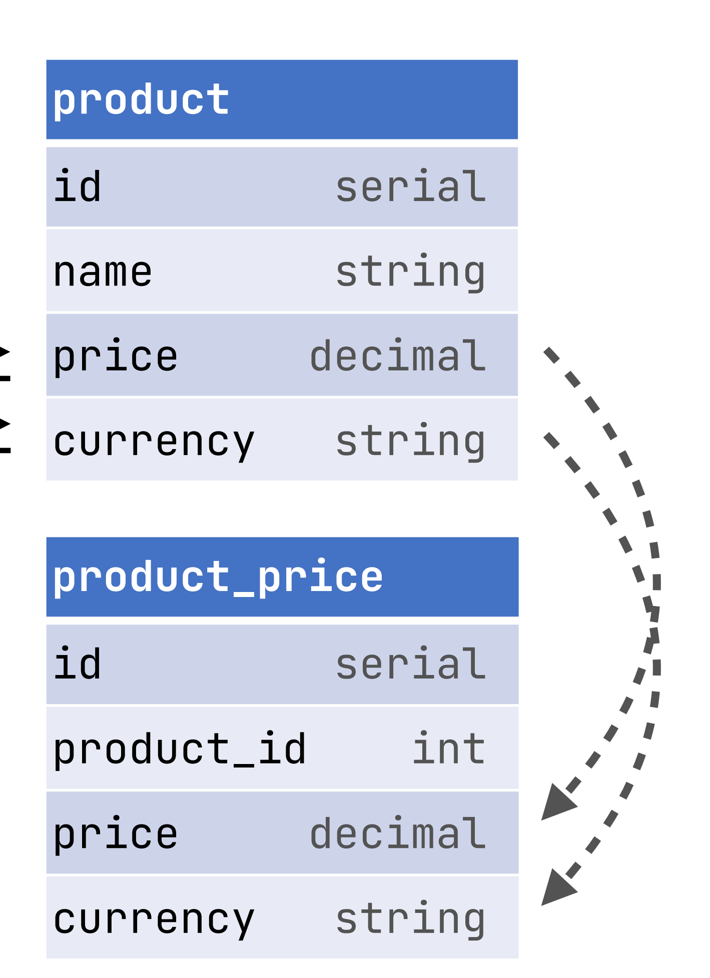
Есть проблемки

1. Да, снижается вероятность локов
2. Но длительность внедрения увеличивается
3. А если внедрение длится час?
4. При этом код еще со старой или уже с новой моделью



Шаг 1. Пишем в обе, читаем из старой

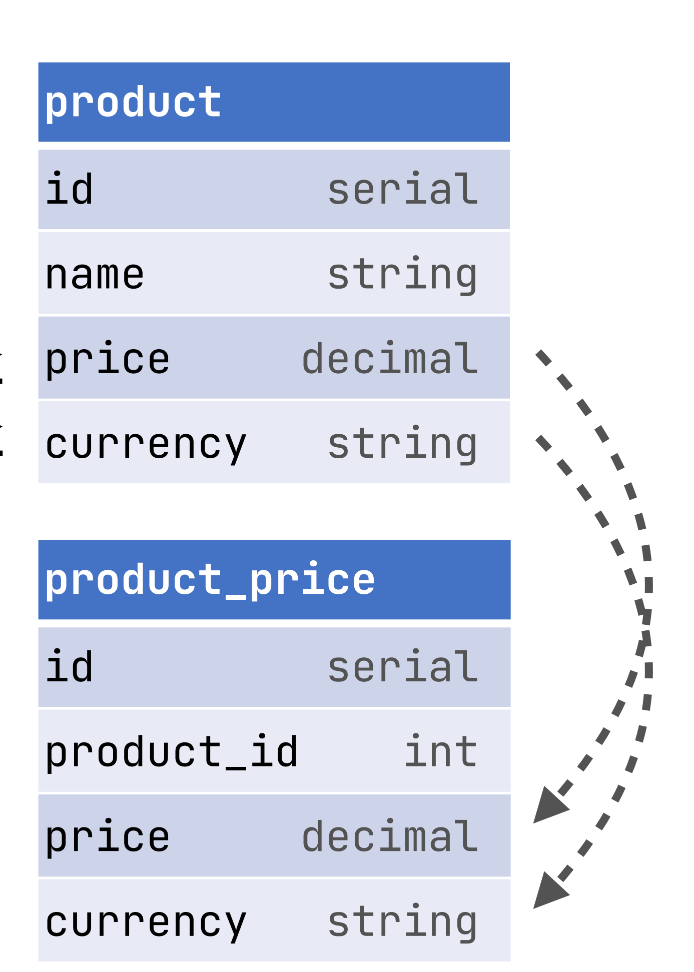
1. **Миграция. Создаем новую таблицу**
2. Пишем в старые поля, дублируем в новые
3. Читаем из старых полей
4. Логика приложения старая (одна цена)





Шаг 1. Пишем в обе, читаем из старой

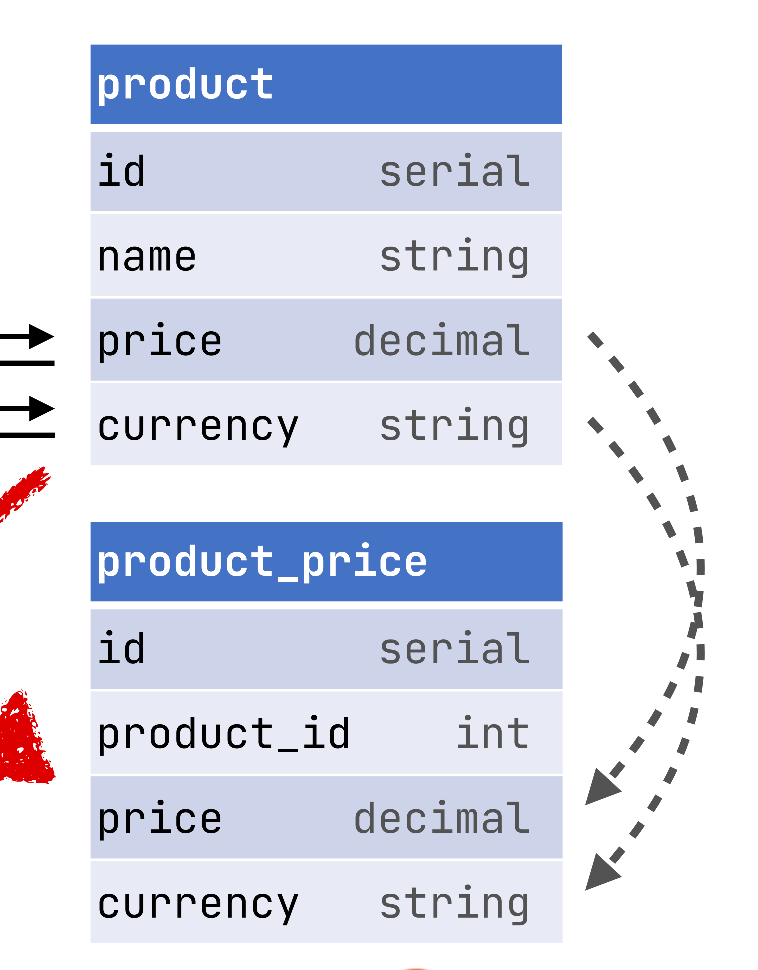
1. Убеждаемся, что правильно заполняется новая таблица
2. Едем дальше





Шаг 2. Пишем в обе, читаем из старой

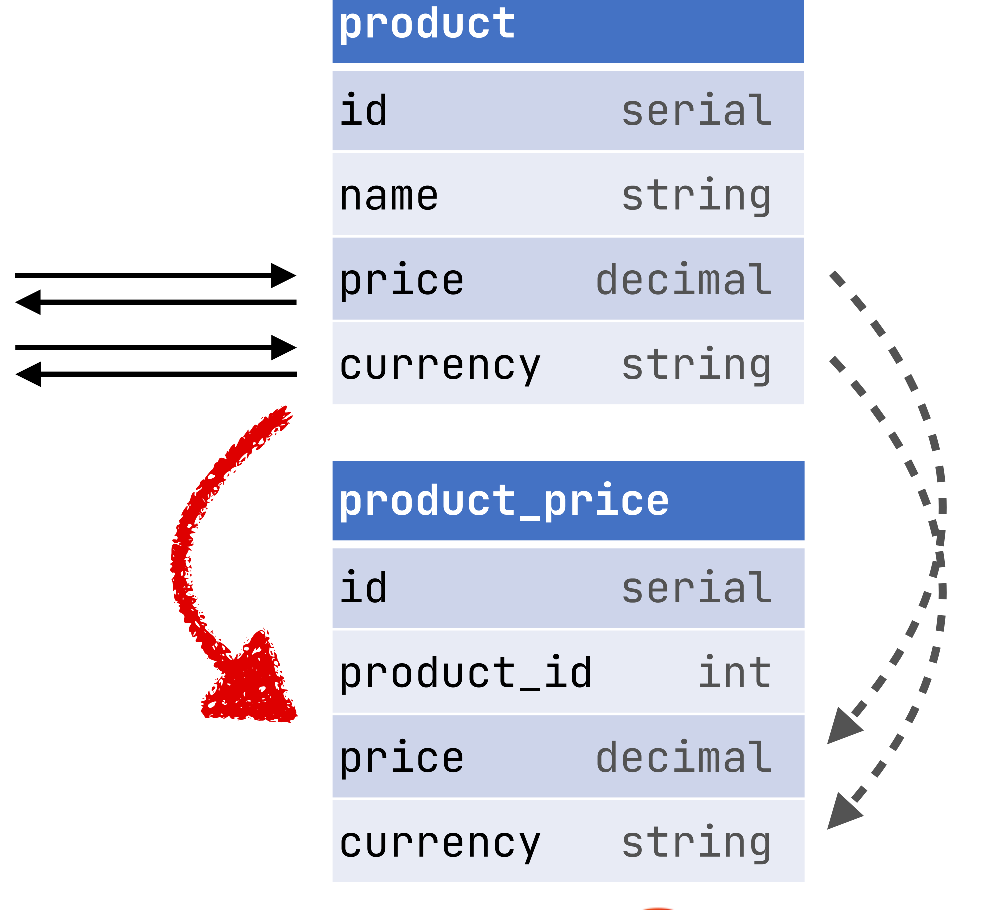
1. **Миграция. Переносим порциями старые данные в новую таблицу**
2. В моделях ничего не трогаем
3. Логика приложения старая (одна цена)





Шаг 2. Пишем в обе, читаем из старой

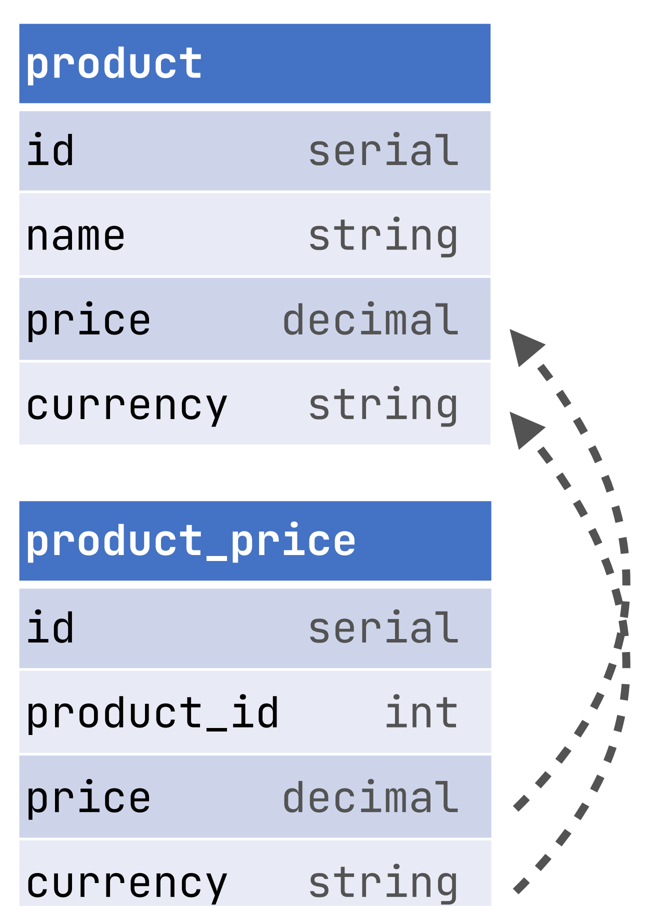
1. Убеждаемся, что новая таблица имеет все записи
2. Едем дальше





Шаг 3. Пишем в обе, читаем из новой

1. **Нет миграций**
2. Пишем в новые поля, дублируем в старые
3. Логика приложения старая (одна цена)

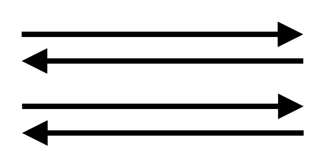


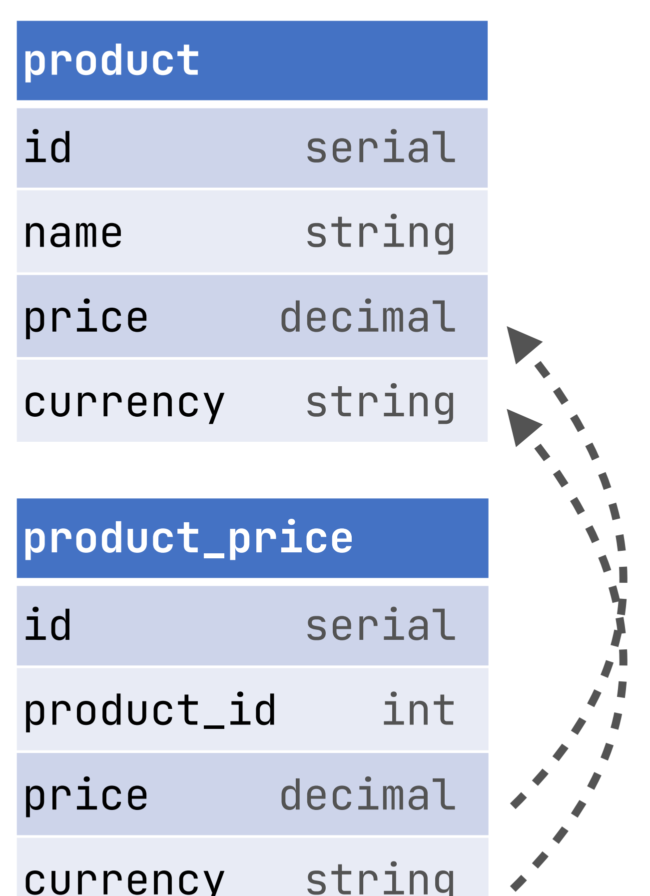


Шаг 3. Пишем в обе, читаем из новой

1. Убеждаемся, что изменения летят в новую таблицу
2. Едем дальше

👍 **С этого и предыдущих шагов есть возможность откатиться**



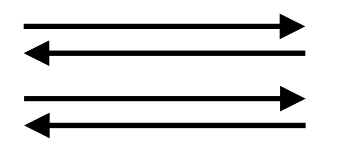


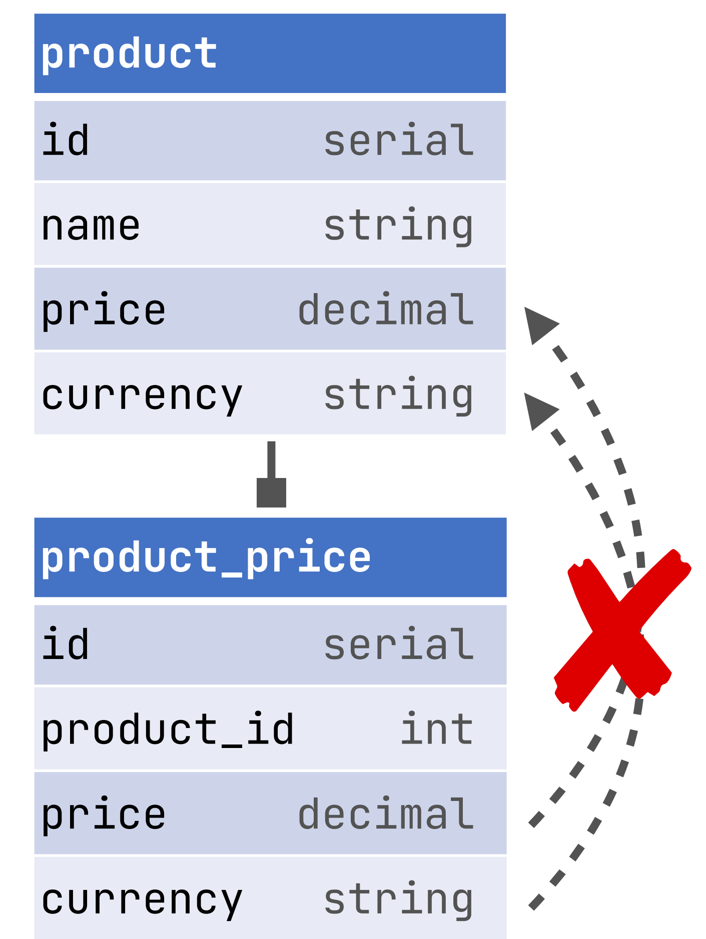


Шаг 4. Открываем фичу множественных цен

1. **Нет миграций**
2. Пишем только в новые поля
3. Логика приложения новая (множественные цены)

👍 **Фича множественных цен работает!**

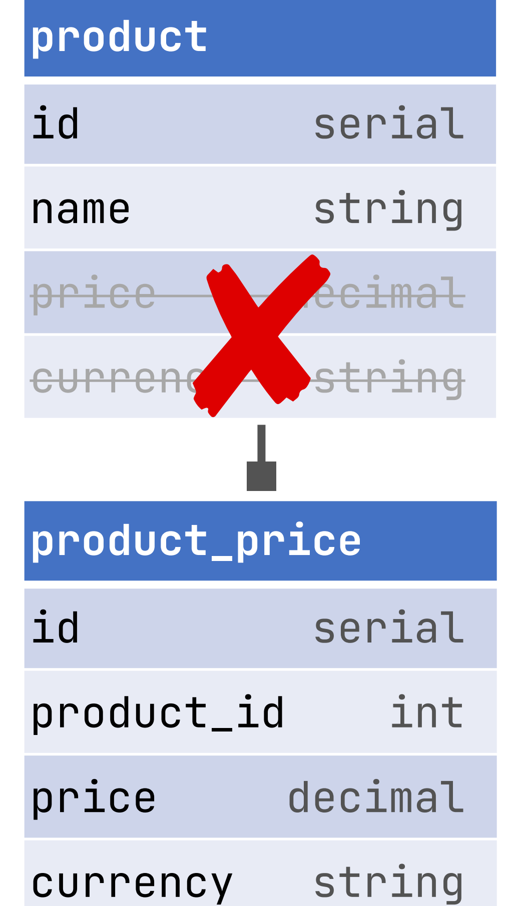


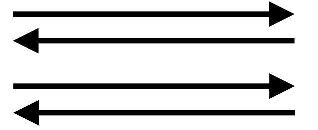




Шаг 5. Подчищаем старое

1. **Миграция. Удаляем поля в product**
2. Удаляем поля в модели
3. Логика приложения новая (множественные цены)







Миграция 1-1 в 1-N. Итого

– Внедрение многоэтапное и длительное

+ Минимизируем локи

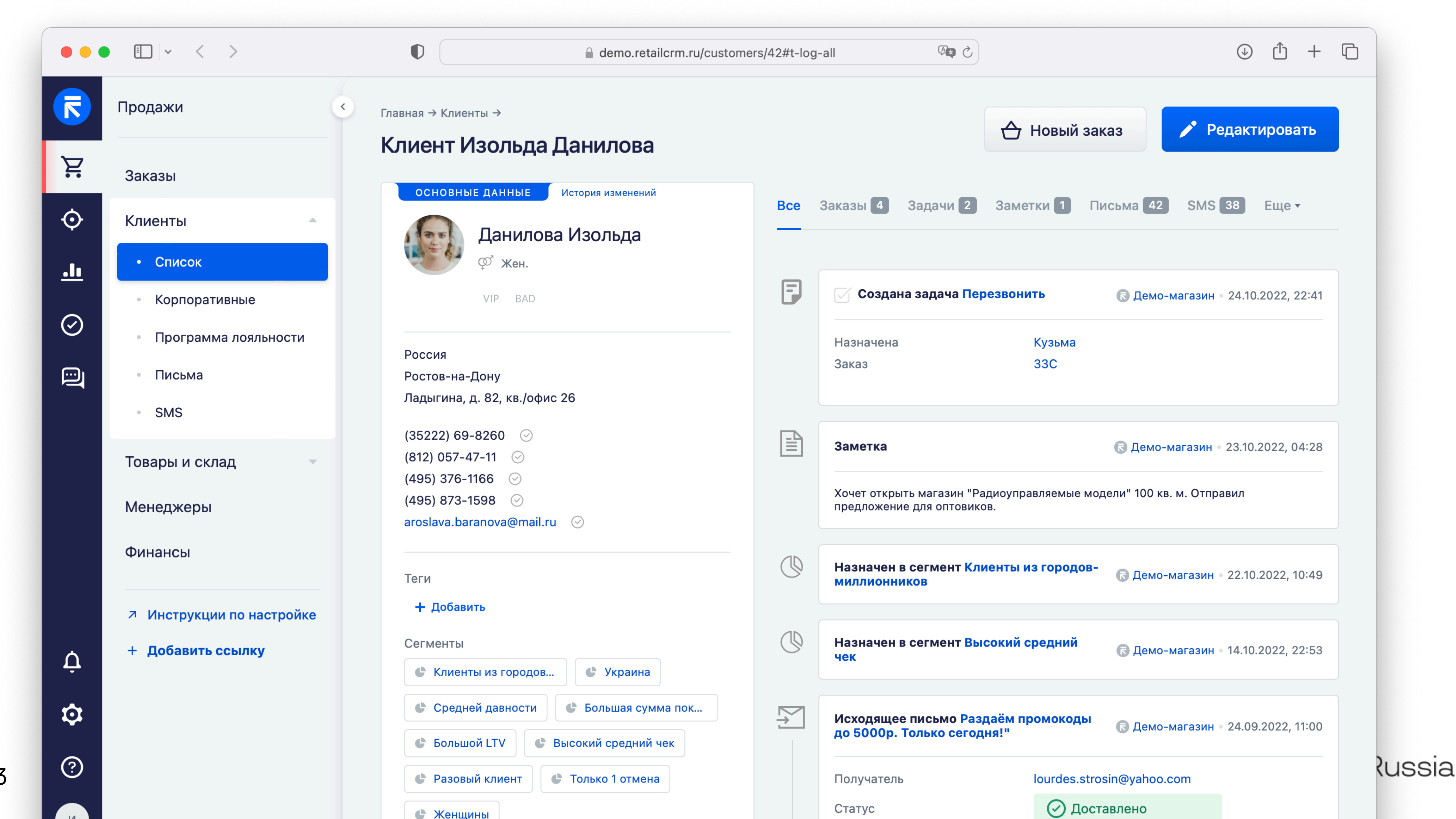
+ Минимально аффектим работу приложения

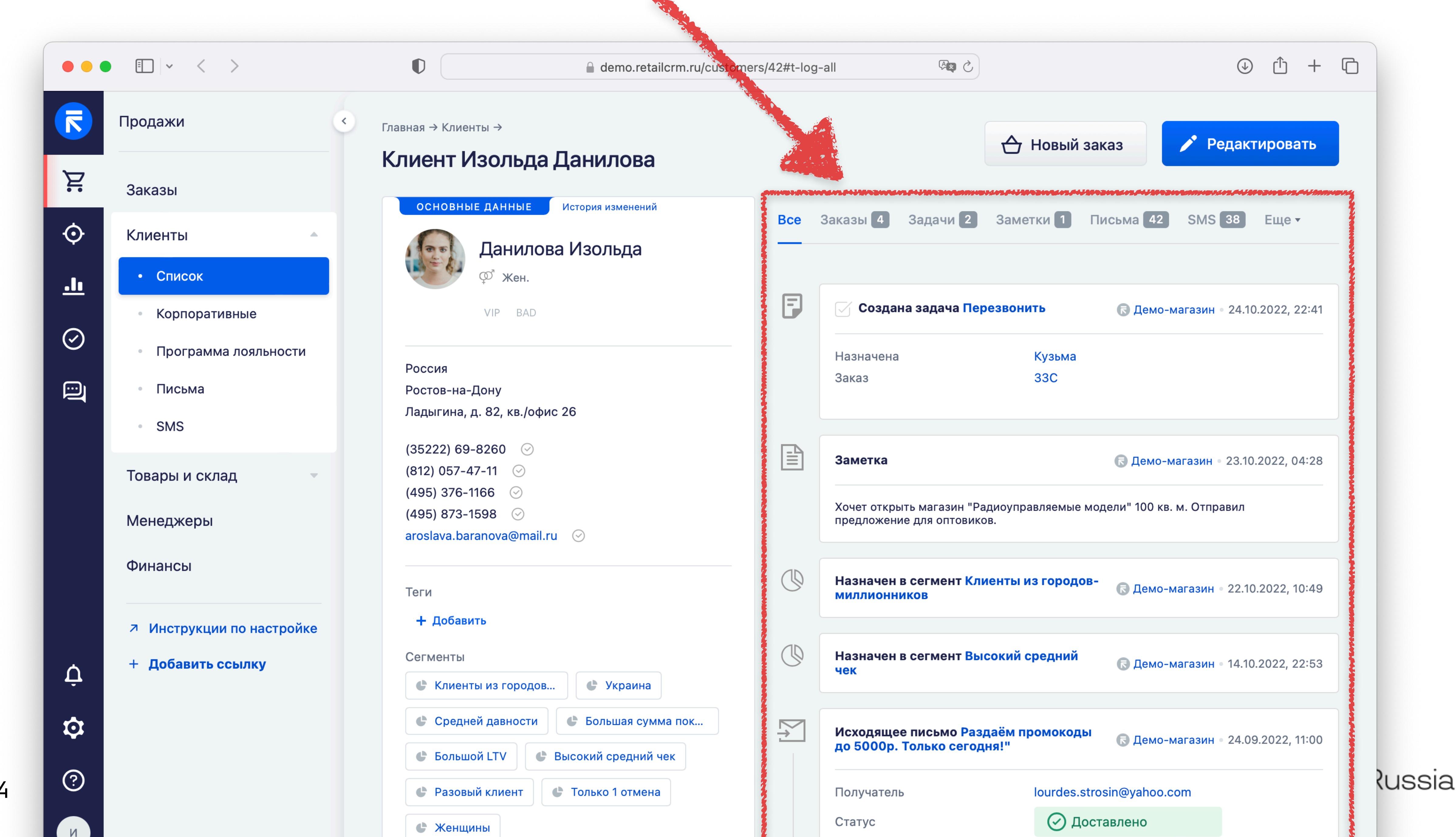
+ В случае ошибок можно откатиться обратно



Индексы




demo.retailcrm.ru/customers/42#t-log-all
Продажи
Заказы
Клиенты
Список
Корпоративные
Программа лояльности
Письма
SMS
Товары и склад
Менеджеры
Финансы
Инструкции по настройке
Добавить ссылку
Главная → Клиенты →
Клиент Изольда Данилова
Новый заказ
Редактировать
ОСНОВНЫЕ ДАННЫЕ
История изменений
Данилова Изольда
Жен.
VIP
BAD
Россия
Ростов-на-Дону
Ладыгина, д. 82, кв./офис 26
(35222) 69-8260
(812) 057-47-11
(495) 376-1166
(495) 873-1598
aroslava.baranova@mail.ru
Теги
Добавить
Сегменты
Клиенты из городов...
Украина
Средней давности
Большая сумма пок...
Большой LTV
Высокий средний чек
Разовый клиент
Только 1 отмена
Женщины
Все
Заказы 4
Задачи 2
Заметки 1
Письма 42
SMS 38
Еще
Создана задача Перезвонить
Демо-магазин · 24.10.2022, 22:41
Назначена
Кузьма
Заказ
33C
Заметка
Демо-магазин · 23.10.2022, 04:28
Хочет открыть магазин "Радиоуправляемые модели" 100 кв. м. Отправил предложение для оптовиков.
Назначен в сегмент Клиенты из городов-миллионников
Демо-магазин · 22.10.2022, 10:49
Назначен в сегмент Высокий средний чек
Демо-магазин · 14.10.2022, 22:53
Исходящее письмо Раздаём промокоды до 5000р. Только сегодня!"
Демо-магазин · 24.09.2022, 11:00
Получатель
lourdes.strosin@yahoo.com
Статус
Доставлено
Russia



demo.retailcrm.ru/customers/42#t-log-all
Продажи
Заказы
Клиенты
Список
Корпоративные
Программа лояльности
Письма
SMS
Товары и склад
Менеджеры
Финансы
Инструкции по настройке
Добавить ссылку
Главная → Клиенты →
Клиент Изольда Данилова
Новый заказ
Редактировать
ОСНОВНЫЕ ДАННЫЕ
История изменений
Данилова Изольда
Жен.
VIP
BAD
Россия
Ростов-на-Дону
Ладыгина, д. 82, кв./офис 26
(35222) 69-8260
(812) 057-47-11
(495) 376-1166
(495) 873-1598
aroslava.baranova@mail.ru
Теги
Добавить
Сегменты
Клиенты из городов...
Украина
Средней давности
Большая сумма пок...
Большой LTV
Высокий средний чек
Разовый клиент
Только 1 отмена
Женщины
Все
Заказы 4
Задачи 2
Заметки 1
Письма 42
SMS 38
Еще
Создана задача Перезвонить
Демо-магазин · 24.10.2022, 22:41
Назначена
Кузьма
Заказ
33C
Заметка
Демо-магазин · 23.10.2022, 04:28
Хочет открыть магазин "Радиоуправляемые модели" 100 кв. м. Отправил предложение для оптовиков.
Назначен в сегмент Клиенты из городов-миллионников
Демо-магазин · 22.10.2022, 10:49
Назначен в сегмент Высокий средний чек
Демо-магазин · 14.10.2022, 22:53
Исходящее письмо Раздаём промокоды до 5000р. Только сегодня!"
Демо-магазин · 24.09.2022, 11:00
Получатель
lourdes.strosin@yahoo.com
Статус
Доставлено

Russia

Лента событий клиента

letter	
id	serial
...	

sms	
id	serial
...	

task	
id	serial
...	

customer_event	
id	serial
letter_id	int
call_id	int
sms_id	int
dialog_id	int
task_id	int
order_id	int
...	
occurred_at	timestamp

call	
id	serial
...	

dialog	
id	serial
...	

order	
id	serial
...	



Автоиндексы от Doctrine

```
phprussia2022=# \d customer_event
...
Indexes:
    "customer_event_pkey1" PRIMARY KEY, btree (id)
    "idx_b64762244525ff26" btree (letter_id)
    "idx_b647622450a89b2c" btree (call_id)
    "idx_b64762245e46c4e2" btree (dialog_id)
    ...
Foreign-key constraints:
    "fk_b64762244525ff26" FOREIGN KEY (letter_id) REFERENCES letter (id) ON DELETE CASCADE
    "fk_b647622450a89b2c" FOREIGN KEY (call_id) REFERENCES call (id) ON DELETE CASCADE
    "fk_b64762245e46c4e2" FOREIGN KEY (dialog_id) REFERENCES mg_dialog (id) ON DELETE CASCADE
    ...
```




Размер индексов

```
phprussia2022=# select count(*) from customer_event;
  count
---------
 3428887
(1 row)

phprussia2022=# ...
   table_name    | table  |  index   |  total
-----------------+--------+----------+---------
 customer_event | 276 MB | 1398 MB | 1674 MB
(1 row)
```

x5 🤯



Структура данных в customer_event

```
phprussia2022=# select * from customer_event limit 9;
   id    | letter_id | sms_id | call_id | dialog_id | task_id | order_id |     occurred_at
---------+-----------+--------+---------+-----------+---------+----------+---------------------
 1540455 |           |        |         |           |    4342 |          | 2018-01-09 12:29:12
 1544286 |           |        |         |           |         |    82909 | 2018-01-11 09:34:29
 1544287 |           | 122452 |         |           |         |          | 2018-01-11 09:34:29
 1544288 |     26631 |        |         |           |         |          | 2018-01-11 09:34:29
 1544289 |           |        |         |           |         |    82358 | 2018-01-11 09:40:53
 1544290 |           | 122453 |         |           |         |          | 2018-01-11 09:40:53
 1544314 |           |        |  304373 |           |         |          | 2018-01-11 10:25:46
 1544318 |           |        |  304379 |           |         |          | 2018-01-11 10:51:13
 1549031 |           |        |         |           |    4438 |          | 2018-01-13 09:08:11
(9 rows)
```

Partial indexes



Partial indexes

```
=# CREATE INDEX idx_b6476224bd5c7e60 ON customer_event (sms_id) WHERE sms_id IS NOT NULL;
```

В индекс добавляются только записи таблицы, подпадающие под условие



Переводим на partial indexes

```
phprussia2022=# \d customer_event
...
Indexes:
    "customer_event_pkey1" PRIMARY KEY, btree (id)
    "idx_b64762244525ff26_notnull" btree (letter_id) WHERE letter_id IS NOT NULL
    "idx_b647622450a89b2c_notnull" btree (call_id) WHERE call_id IS NOT NULL
    "idx_b64762245e46c4e2_notnull" btree (dialog_id) WHERE dialog_id IS NOT NULL
    ...
Foreign-key constraints:
    "fk_b64762244525ff26" FOREIGN KEY (letter_id) REFERENCES letter (id) ON DELETE CASCADE
    "fk_b647622450a89b2c" FOREIGN KEY (call_id) REFERENCES call (id) ON DELETE CASCADE
    "fk_b64762245e46c4e2" FOREIGN KEY (dialog_id) REFERENCES mg_dialog (id) ON DELETE CASCADE
    ...
```




Обновленный размер индексов

```
phprussia2022=# select count(*) from customer_event;
  count
---------
 3428887
(1 row)

phprussia2022=# ...
   table_name    | table  |  index  |  total
-----------------+--------+---------+---------
 customer_event | 276 MB | 428 MB | 704 MB
(1 row)
```

экономия x3 👍



На уровне Doctrine

```yaml
services:
    App\DBAL\Schema\IndexNullsChanger:
       arguments:
            - '@annotation_reader'
        tags:
            - { name: doctrine.event_listener, event: postGenerateSchemaTable }
```




Собственные partial indexes

```
/**
 * @ORM\Table(
 *     name="tag",
 *     indexes={
 *         @ORM\Index(
 *             name="tag_name_deleted_at_isnull_ind",
 *             columns={"name"},
 *             options={"where"="deleted_at IS NULL"}
 *         ),
 *     }
 * )
 *
 * @ORM\Entity(repositoryClass="TagRepository::class")
*/
```




Но если хочется большего? 



Индексы на выражения

```
CREATE INDEX i_cindex_b144d4115c806fc66934701c1fe15fa1 ON product (lower(name));
```




Индексы на выражения

```
=# EXPLAIN ANALYZE SELECT * FROM product WHERE lower(name) = 'd9aa6210ac590cb9575d83e2d30b86c1';
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on product  (cost=20.29..783.05 rows=500 width=37) (actual time=2.219..2.223 rows=1 loops=1)
   Recheck Cond: (lower((name)::text) = 'd9aa6210ac590cb9575d83e2d30b86c1'::text)
   Heap Blocks: exact=1
   ->  Bitmap Index Scan on i_cindex_b144d4115c806fc66934701c1fe15fa1  (cost=0.00..20.17 rows=500 width=0)
         Index Cond: (lower((name)::text) = 'd9aa6210ac590cb9575d83e2d30b86c1'::text)
 Planning Time: 1.577 ms
 Execution Time: 2.263 ms
```




Другие типы индексов

```
CREATE INDEX
  i_cindex_e71bccfe2282f815a701c1c8e95da679
ON product USING gin (lower(name) gin_trgm_ops);
```




Другие типы индексов

```
=# EXPLAIN ANALYZE SELECT * FROM product WHERE lower(name) LIKE '%210ac590cb9575d83e%';
                                                     QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on product  (cost=196.08..232.94 rows=10 width=37) (actual time=2.131..2.132 rows=1 loops=1)
   Recheck Cond: (lower((name)::text) ~~ '%210ac590cb9575d83e%'::text)
   Heap Blocks: exact=1
   ->  Bitmap Index Scan on i_cindex_e71bccfe2282f815a701c1c8e95da679  (cost=0.00..196.07 rows=10 width=0)
         Index Cond: (lower((name)::text) ~~ '%210ac590cb9575d83e%'::text)
 Planning Time: 1.038 ms
 Execution Time: 2.854 ms
```




Но Doctrine ORM не умеет 🥲



intaro/custom-index-bundle



Описание индексов в модели

```
use App\Repository\ProductRepository;
use Doctrine\ORM\Mapping as ORM;
use Intaro\CustomIndexBundle\Annotations as CI;

/**
 * @ORM\Entity(repositoryClass=ProductRepository::class)
 * @CI\CustomIndexes(indexes={
 *     @CI\CustomIndex(columns="lower(name)"),
 *     @CI\CustomIndex(columns="lower(name) gin_trgm_ops", using="gin"),
 * })
 */
class Product
{
    // ...
}
```

Создание индексов в БД

```
➜  bin/console intaro:doctrine:index:update
No index was dropped.
Index i_cindex_e71bccfe2282f815a701c1c8e95da679 was created.
```




Summary



Что в итоге

1. Эффективно подключаемся к PostgreSQL



Что в итоге

1. Эффективно подключаемся к PostgreSQL
2. Контролируем время выполнения запросов и время на локи



Что в итоге

1. Эффективно подключаемся к PostgreSQL
2. Контролируем время выполнения запросов и время на локи
3. Управляем структурой БД через модель данных



Что в итоге

1. Эффективно подключаемся к PostgreSQL
2. Контролируем время выполнения запросов и время на локи
3. Управляем структурой БД через модель данных
4. Следим за синхронностью модели и схемы данных



Что в итоге

1. Эффективно подключаемся к PostgreSQL
2. Контролируем время выполнения запросов и время на локи
3. Управляем структурой БД через модель данных
4. Следим за синхронностью модели и схемы данных
5. Внедряем «тяжелые» фичи, не влияя на работу пользователей



Что в итоге

1. Эффективно подключаемся к PostgreSQL
2. Контролируем время выполнения запросов и время на локи
3. Управляем структурой БД через модель данных
4. Следим за синхронностью модели и схемы данных
5. Внедряем «тяжелые» фичи, не влияя на работу пользователей
6. Строим оптимальные индексы



Спасибо! Вопросы?

Проголосуйте за доклад :)



Ильяс Салихов

tg @salikhov_ilyas

simla.tech



PHP Russia
2022